The Pathway for Agent and BOT Enemy in Strategy Game in Multilayer with High Efficiency Model

Khammapun Khantanapoka[1], Suwanna Rasmequan[2]
Department of Computer Science, Faculty of Science, Burapha University, Thailand.
(e-mail: doramon86@hotmail.com[1])

***Abstract*— The path finding analysis has importance for various working such as logistics, transportation, operation management, system analysis and design, project management, network and production line. Especially, game programming technology has effect to economic and dominates culture, increasingly. The shortest path analysis is artificial intelligent which developed capability about think cause and effect, learning and thinking like human. Heuristic technique is method for solving a problem which gets result or not. It may return unexpected value which depends on each problem. Some 2D&3D strategy games kind adventure style uses shortest path analysis control path for movement of character, wheeled vehicle, animals, and fantasy hero. Such as SWAT, diabo games, commandos it has character and controls semilar to strategy games (click and movement) combine first Personal shooting games or FPS (movement in building). It does not focus to build the construction. The strategy gameuses path for movement ever time, both in building, terra and overground. The position of 3D object use refers 2D coordinate. Generally, the most of game on LAN and online will move on terrain and inside building or the room.**

This research proposes three essences.

First, this research proposes Depth Direction A* algorithm method. A new method uses linear graph theory cooperates with A* classic Algorithm calculates efficiency of avoid obstacles object on maps and search for shortest path of multi-layer. It decrease expand child node of A* effect to speed process rapidly. The child node grows less than A* algorithm in clear area and plan about expand node before processed. It guarantee about expand node less than A* algorithm node certainly. It takes time movement and avoid obstacles object less than uses other algorithm.

Second, the position of each players or AI enemy (BOT) transfers on multilayer with perspective projection Method. For players, it can set beforehand of pathway with click mouse for kept begin position and destination position as same as 2D strategy games. Both begin and destination may be difference class (layer) in the building. For BOT, it can find shortest path to duties quickly. The system of game will sends begin position and destination position which is difference layer with Perspective Projection Method(PPM). It is intelligent enough to acknowledge difference size in each class.

Third, abstract graph of each pattarn depend terrain type.It use movement by does not expend child node of octiles and return shortest path or optimum path.

The whole solution base on decrease expend child node and take time are extremely while increase efficiency in path finding quickly and smooth pathway.

This research work able decrease expand child node in path finding process 22.12-70.67% depend on property of terrain and object propeterty by use abstract graph from pathway database together with DDA* algorithm.

***Keyword*— path finding, perspective projection method.**

I. INTRODUCTION

The most algorithms used path planning expanded child node around agent to octiles for next position analysis. The agents mean an object controlled movement by player or Bot enermy.The hard obstacles means octiles agent cannot traversal capability. The several algorithms development in path planning uses idea pattern give artificial intelligence solves a problem in one octiles. (eight child nodes around tile.) It work repeat a pattern until recent node same as destination. There are several algorithms improved from A* classic such as HPA* build an abstract search graph by dividing the environment into square clusters connected by entrances and finding the shortest path between them, PRA* builds a multi-level search space by abstraction cliques of node. The results is to narrow the search space in the original problem to a window of node along the optimal shortest path. Both HPA* and PRA* are focused on solve planning problems for homogeneous agents in homogeneous terrain environments.[1] However, A* algorithm has two important process. First, A* star algorithm measure distance between begin node to destination node. Second, A* algorithm expended child node around current node and comparison cost value for deciding change position.

This research address base on hypothesis; if two important process of A* algorithm improved, It has more efficiency then another algorithm which improve base on A* algorithm.

Fig.1. Example of strategy game: strategy game kind adventure style. http://pcmedia.ign.com/pc/image/article/918/918735/diablo-iii-20081010003738560.jpg,http://www.frictionlessinsight.com/revpics/SilentStorm/SilentStormC.jpg,http://img.gamespot.com/gamespot/images/2003/news/07/08/laser/squad_screen007.jpg ,http://i.d.com.com/i/dl/media/dlimage/79/67/7967large.jpeg

The long pathways will non-smooth in clue way of slope because does not consider overall pathways. If pathways want natural must to separate consider the very part which uses the large size of memory and effect to take time processing very much. The most research try search for new method which solve this problem. It must equilibrium extremely between natural pathway and takes time at less that it is difficult possible. In this paper, the researcher try to improve new method base on A* classic to advantage for every application which developed from A* classic principle***. Our idea is Depth Direction A*(DDA*).***

Beside, Some games has multilayer such as object can move in the lake, rever and terrain (overcraft) and in the building that each layer has cost value and obstacles object are difference. The strategy games have rendering process which using takes time very much. The experiment must define rendering time are stable because want measure efficiency of path finding algorithm by independency.

II. ALGORITHM DETIAL

A. Analysis of A-Star(A) classic Algorithm*

A* is graph search algorithm that finds the least-cost path from a given start node to one goal node (out of one or more possible goals). It uses a distance-plus-cost heuristic function (usually denoted f(x)) to determine the order to search visits nodes in the tree graph. The distance plus-cost heuristic is a sum of two functions: the path-cost function (usually denoted g(x), which may or may not be a heuristic) and an admissible heuristic estimate of the distance to the goal.The path-cost function g(x) is the cost from the starting node to the current node.

A-Star Algorithm Pseudo Code

Create Start Node with Current Position
Add Start Node to Queue
    While Queue Not Empty
    Sort Node Queue by f(N) Value in Ascending
    Get First Node From Queue call Node “N”
    If N is Goal Then Found and Exit Loop
    Else
    Mark N Node as Visited
Expand each reachable Node from N call Node “Next N”
f(Next N) = g(Next N) + h(Next N)
Loop

III. EXPERIMENT AND RESULT

A. Weak perspective projection model

The weak perspective model defines a geometric mapping, which stand between the perspective and affine models. This model considers that distance between points in the scene is small relative to focal length as follow

$$y_i = \frac{y_p}{\mu_z} f \qquad \text{and} \qquad x_i = \frac{x_p}{\mu_z} f \qquad (1)$$

Where μ_z is the average *z* coordinating of all the point in a scene.

It illustrates two possible geometric interpretations for relationships define in Equation 1. Illustrates a two step process wherein, first, all point are affine projected to a plane orthogonal to the image scene plane and at a distance μ_z. The server send position 2D array in small plane uses calculate join in center of projection and filter scaled position which is filter encoded on the client. These processes decoded position value on 3D real scene and layer of scene to another client.

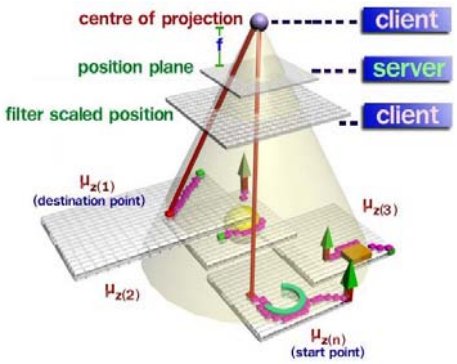


Fig.2. Weak perspective projection model

It sent start position and destination position to the layer of scene. The next step use AI of path finding search for pathway between two points and between layers. Point on this plane is then mapped into the image scene plane by a perspective projection. The projection on the scene plane $z = \mu_z$ simply replaces the Z coordinates of the point by μ_z .Since points are assumed to be close; this projection is a good approximation of the scene. The weak perspective method corresponds to a perspective model for scenes approximated by plane parallels to the image scene plane. A second geometric interpretation of this equation to show in figure, we can combine the value f and μ_z into the single constant. This equation corresponds to a scale on scene plane. The perspective position model uses the algebra of the projective space points are mapped into an image scene plane, which mapping can also be defined using Euclidean transformations. It using homogeneous coordinates accompany geometry of image scene formation is simply defined by the projection of a three dimension point into the scene plane. The matrix *H* is not square because a point in a higher dimension is map into a lower dimension perspective position model is defined by a projection transformation. For scene in multilayer which has size are not same. The method used special case of affine model which scale position, first, object in scene are mapped into the scene image plane by an affine projection and there the position is rescaled by a value f / μ_z .The affine model can be seen as a particular case of the perspective model when $f / \mu_z = 1$ .The perspective model can be formulated by changing the projection equations of the perspective or the affine models follow requirement, include a change in scale for client. The change filter scaled position use update version of game.

$$Q_A = \begin{bmatrix} f/\mu_z & 0 & 0 & 0 \\ 0 & f/\mu_z & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig.3. This figure show scale perspective for projection transformation.

The scene size can scale use move the scale factor in this metric;

$$P = VQ_A M_A \tag{2}$$

or expression:

$$P = \begin{bmatrix} s_u & s_u \cot(\varphi) & s_0 \\ 0 & s_v \sin(\varphi) & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fig.4. The matrix expressed of weak perspective projection model.

where $s_u = fk_u / \mu_z$ s and $s_v = fk_v / \mu_z$ The length to the scene plane of each layers $P = VQ_A M_A + (\omega L_{layer})$ by ω is difference between each layer and L is number of layer.The weak perspective is a scaled version of the affine model. The scale is a function the f that defines the distance from the centre of the camera to the image plane and the average distance μ_z .This source code is in the technical report support this paper which implement by matlab because is matrix data.

B. Depth Direction A (This research proposes)*

This research proposes Depth Direction A* Algorithm method which new method which use linear graph theory cooperate with A* classic Algorithm for shortest path of multi-layer quickly.

Depth Direction A-Star Algorithm Pseudo Code

Create Start Node with Current Position
Create A Linear Function by m = (Start Y – Goal Y) / (Start X – Goal X)
Add Start Node to Queue
    While Queue Not Empty
    Sort Node Queue by f(N) Value in Ascending
Get First Node From Queue call Node “N”
    If N is Goal Then Found and Exit Loop
    Else
    Mark N Node as Visited
Expand a Node by Linear Function call Node “Next N”
    If Found Obstructions Then
    f(Next N) = g(Next N) + h(Next N)
    adjust Algorithm Style to A-Star Algorithm
    Loop

C. The algorithm of Path finding in each layer.

From experiment, the first of step, there are compare between seven type of path finding algorithm sure get algorithm which expand node and take time are least when working alone. This experiment use multilayer terrain by one layer, three layers and nine layers. It shows result of working clearly as follows: Dijkstra’s algorithm, Best First Search, Depth First Search, Iterative Deepening algorithm, A-Star Algorithm (A*), Iterative Deepening A* algorithm, Breadth First Search algorithm.

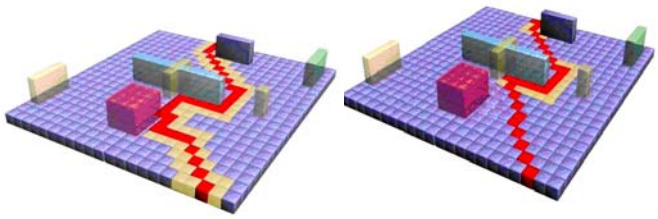
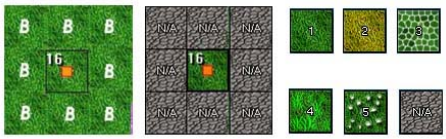


Fig. 5. This figure show trace expend node (a) A* classic and (b) DDA*

The A* classic Algorithm has pathway as non-smooth. This paper solve a problem using Linear Direction graph theory cooperate with A-Star (A*) Algorithm. It use A* Algorithm for some situation when avoid obstacles object, it use Linear Direction graph theory for some situation which increase speed while decrease expand child node of A* classic.

Where *m* as slope of straight line:

$$m_{layer} = \frac{y_1 - y_2}{x_1 - x_2} \tag{2}$$



(a) (b) (c)

Fig. 6. From Experiment (a) Around of current node have not obstacles every side.(AND), (b) Around of current node at least one side close obstacles which weight extremely and can not pass certainly (OR), (c) the kind of obstacles which are weight value different.

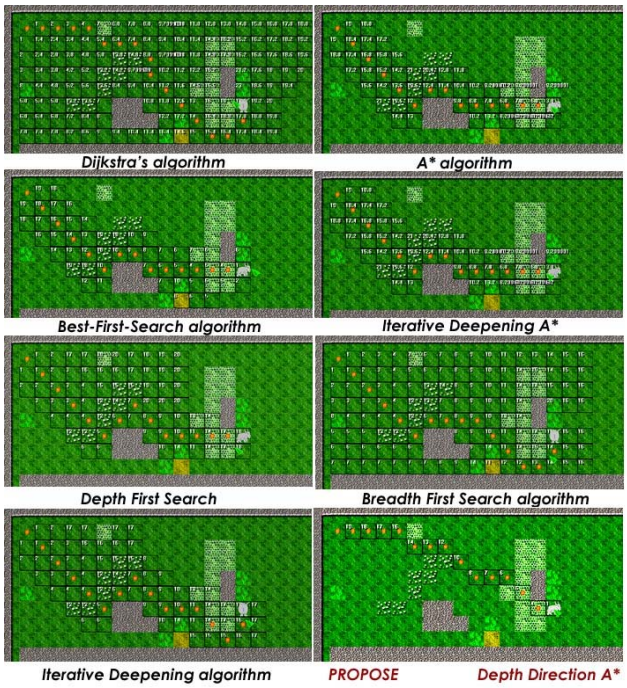


Fig.7. Pathway of each algorithm which difference about rapidity, expand node, intelligence.

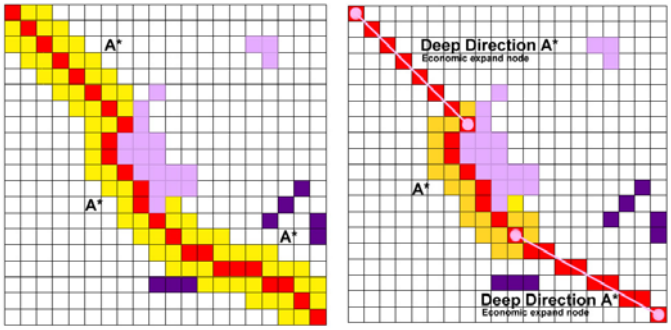


Fig.8. The trace calculate and expend node (a) A* classic and (b) DDA* algorithm (The Yellow block is clue of expend child node.)

D. *Using DDA* in strategy Game kind adventure style with dynamic environment (This research proposes)*

If current node is not close any obstacles every side use calculates Linear Direction graph promptly and when current node has at least one side close any side of obstacles will switch use A* algorithm. When current node is not close any obstacles every side switch use calculates Linear Direction graph again. It applies to multilayer of path finding. It switch are cycle until agent to destination position. We are expanding to work with three dimensions or multilayer.

The first of case: each class has a ladder connected between two classes called “single transition layers”. The class has not begin position or destination position will manage for movement by artificial intelligence (AI). Each class define sub-begin position and sub-destination position is connected point between two class in building such as, ladder, lift which using position for change from a class to another class called Y-axle (Change layer of Y-axle or CLY) and fix sub-destination of that class. In each class have CLY from other class and CLY to the next class. The class address begin position by players will find and traverse with depth direction A* algorithm to CLY which has sub-begin for next class. The class address destination position by players will find and traverse from CLY to destination position.

Other class will manage by AI which use Depth Direction A* algorithm. The pathway of AI will recall in loading step standard pathway or reloaded file saved and repeat generate abstraction graph every times when increase new hard obstacles in that class. It is dynamic operation.

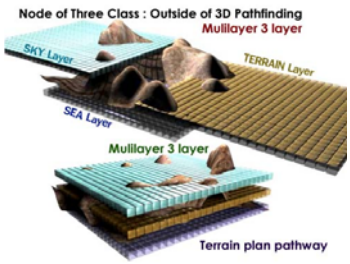


Fig.9. In case: each class of waterfall. Node of terrain multilayer (Three layers) which can change layer in Y-axle (Y-layer_1,Y-layer_2,Y-layer_3) with X-axle and Z-axle of position in lower layer or higher layer which must are not hard obstacles.

Sub-Case 1: The ladder of every class same Y-axle (change layer Y-axle) such as lift or ladder by calculates with equation as follow:

$$\text{SV}_{node} = \sum_{l=1}^{layer}\sum_{i=1}^{n}\left(\frac{Y1-Y2}{X1-X2}\right)_i + (g(n+h(n))_i \quad \textbf{(2)}$$

$$\text{TSV}_{node} = \text{SV}_{\text{node}} + \text{CLY}_{\text{n}} \quad \textbf{(3)}$$

where

SV as amount of node every class (2 layer).

$\boldsymbol{CLY_i}$ as amount node of change layer in each class.

n as amount of layer.

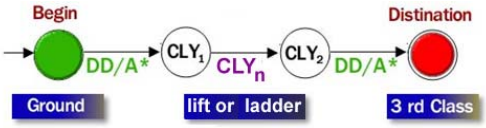


Fig.10. The state transition of pathway is single transition layer.

Sub-Case 2: The ladder is not same Y-axle (change layer Y-axle) calculate with equation as follow:

$$\text{TSV}_{node} = \sum_{l=1}^{layer}\sum_{i=1}^{n}\left(\frac{Y1-Y2}{X1-X2}\right)_i + (g(n+h(n))_i + CLY_i \quad (4)$$

where

TSV as amount of node every class (All layer).

$\boldsymbol{CLY_i}$ as amount of node of change layer in each class.

n as amount of switch DDA*/A* all layer.

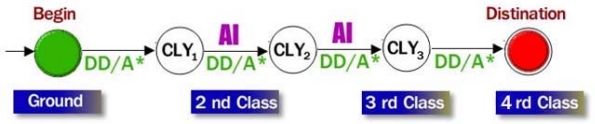
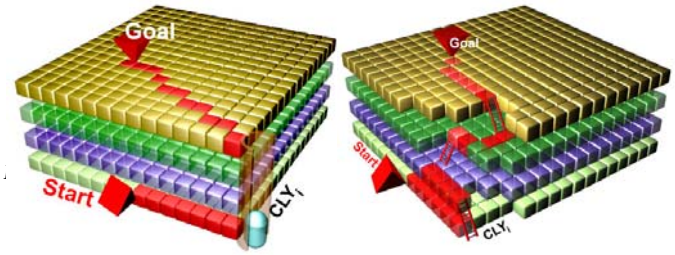


Fig.11.The state transition stays within buildings which example has 4 classes.



(a) (b)

Fig.12. These are comparison between (a) single transition layer and (b) multiple transition layer.

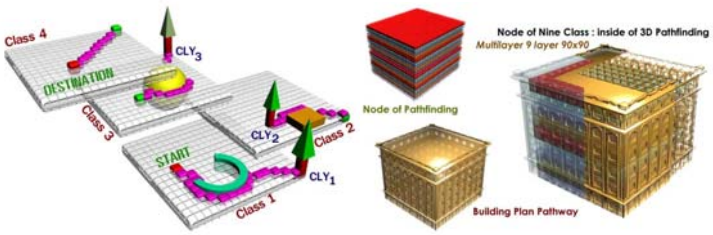


Fig.13.The multilayer of building has nine layers which can change layer specifically position of change layer such as ladder, case in does not use lift will define value change of layer are stable (Change Layer in Y-axle: CLY_i)

When define position for transition layer travease from a layer to close layer such as ladder, lift in building. Example, begin point is ground floor and destination position is in 3rd class. First of path, begin point on ground to ladder#1. Second are path from ladder#1 to ladder#2 on the 2nd class. Third of path, define from ladder#2 on second class to destination point in 3rd class. In sub-case 1, agent does not traverse in second class. Because lift pass 2nd class are non stop. In sub-case 2, agent does traverse in second class by using depth direction A*.

This paper propose separate calculation pathway with each class before altogether. It is easies to update abstract graph each class when adding new hard obstacles. In this experiment define weight of transition layers for one class is stable (Change Layer in Y-axle: (CLY_i=1).

The first of case: each class has ladder or transition layers than a position in a class called “multiple transition layers”.

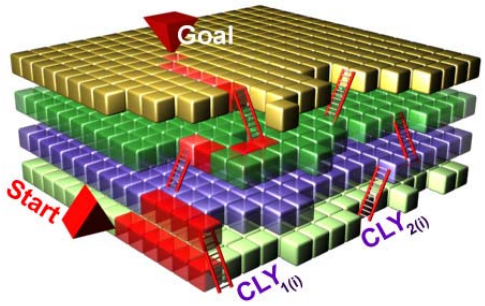


Fig.14. The state transition of pathway which has multiple transition layers.

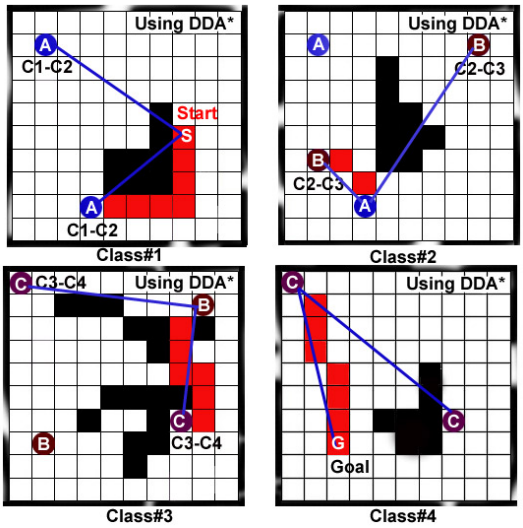


Fig.15 Example of comparisons between start octiles with two transition layers in first class, it effect to path in second class and next class.

The abstract graph mean whole pathway which are possible. The whole pathways keep to pathway database. It will update information when each structure pathway to change. From figure 15. shows agent traverse in the building. In class#1, the begin position compare with distance of two transition layers which connected class#2 and choose a pathway for below (A) which are shortest path. It using DDA* in class#1.This step expand child node for found next position of octiles .When agent up to Class#2, it retrieve abstract graph form database and choose a pathway for below (B) which are shortest path. This step does not expand child node for found next position of octiles and using pathway from database. When agent up to Class#3, it uses operate same as Class#2. In class#4, the agents traverse from transition layer point to (G) destination position by DDA*. All operation bases on principle of decrease expand child node in A* for operate quickly. So, our work will expand child node specific only class addresses begin position and destination.

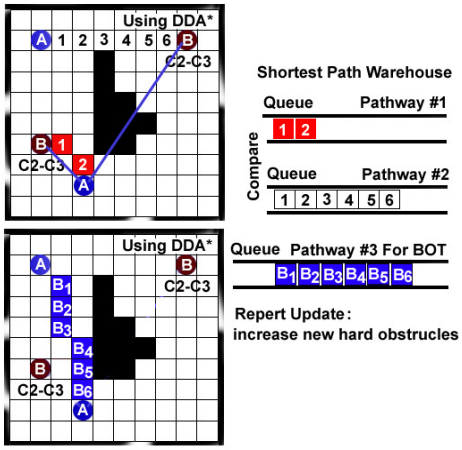


Fig.16 Example of pathway database, (Top) comparison distance between transition layer which both are connected to class#3.(Under) Path connected transition layer in same class which using for movement of BOT enemy.

.

It keeps every pathway are possible in each class to pathway database in abstract graph format.[3] It keeps path inside class and position of transition layers. Beside,It keeps path of harbour in each sea which connected. It represent a map and used player or BOT enemy. The system of game choose optimal pathway from data of agent position. It deciding chooses CLY_i cause distance between begin position to CLY_i and destination position to CLY_i that nearest and shortest.

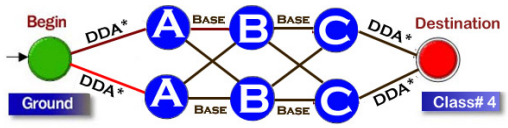


Fig.17. The pathway of multiple transition layers use DDA* together with pathway database which keep information of abstract graph.

E. Sub-Case 3: each class has size are differences and connected scatteredly.

The size of each class are difference depend on shape of building. Some class connect disorderliness such as both Class#1 and Class#2 has transition layer point connect to Class#4,Class#1 has transition layer point connect to Class#2 and Class#3.It connects by free. The pathway finding process valuated all possible pathways before time. It helps return optimum pathway and quickly.

Fig.18. Example of each class depend on shape of building, (left) http://media.bestofmicro.com/nature-vs-hollywood,I-0-161352-13.jpg (right) http://www.gearthblog.com/images/images307/berlin.jpg

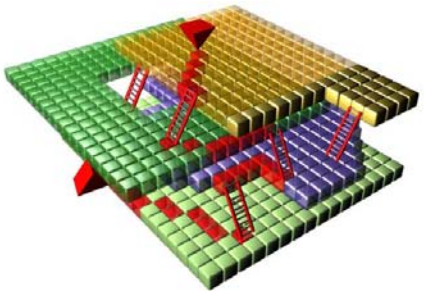


Fig.19. The each class has size are differences and connected scatteredly.

From example, we define weight of use each transition layer point is equal to 1. We can enumerate pathways which are connected disorderliness as follow

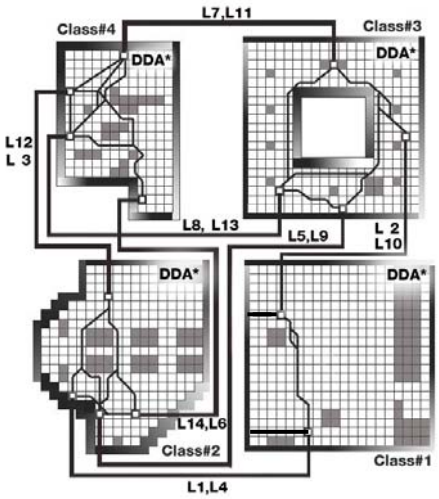


Fig.20. The state transition of pathway which has multiple transition layers.

The each class will connected with multiple layers. The first of step, beginning considers each transition layer of Class# 1. It connected to Class#2 as link L1 and Class#3 as L2. The second of step, we consider each transition layer of Class# 2. It connect to Class#4 as link L3 , Class#1 as link L4, Class#3 as link L5 and Class#4 as L6. It found L1 and L4 overlap each other. We choose only a pathway and keep to pathway database which are L1. From Table 18, it shows overlap analysis of each pathway. We considers until up to Class#4 and return all pathway are possible. Third, we get only route is necessary. This is abstract graph for automatic controls around movement of hero or BOT enemy. It fixed transition layers. From figure 22 shows each class define optimal pathway for that class by DDA* algorithm. In Class#1 connected between transitions layers to each door.

Link	Connected	Transition Layers	Compare Link	Choose Path Storage
L 1	Class#1 ➔ Class#2	(C1) (C2)	***L 2 - L10***	L 2
L 2	Class#1 ➔ Class#3	(C1) (C3)	***L 1 – L 4***	L 1
L 3	Class#2 ➔ Class#4	(C2) (C4)	***L 3- L 12***	L3
L 4	Class#2 ➔ Class#1	(C2) (C1)	***L 6 –L 14***	L 6
L 5	Class#2 ➔ Class#3	(C2) (C3)	***L 7 –L 12***	L 7
L 6	Class#2 ➔ Class#4	(C2) (C4)	***L 8 –L 13***	L 8
L 7	Class#3 ➔ Class#4	(C3) (C4)	L 5	L 5
L 8	Class#3 ➔ Class#4	(C3) (C4)	L 9	L 9
L 9	Class#3 ➔ Class#2	(C3) (C2)	L 11	L 11
L 10	Class#3 ➔ Class#1	(C3) (C1)		
L 11	Class#4 ➔ Class#3	(C4) (C3)		
L 12	Class#4 ➔ Class#2	(C4) (C2)		
L 13	Class#4 ➔ Class#3	(C4) (C3)		
L 14	Class#4 ➔ Class#2	(C4) (C2)		

Fig.21. Example of table show connection of multiple transition layers. The gray colors cells duplicate two pathways by choose only one pathway push to pathway database.

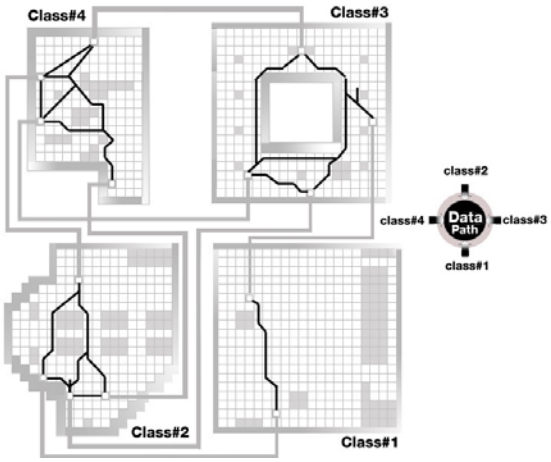


Fig.22. This is shortest path process of inside building.(a) Multiple transition point class#1–2-3-4 which use DDA* find and keep that pathways.

F. The auto-transition terrain operation in strategy game.

The most strategy game kind adventure style has operation of transportation in harbour controled by player such as commandand conquer redalert version 3 but this research propose shotest path using auto-transition terrain. It gets shotest path using harbour automatically.

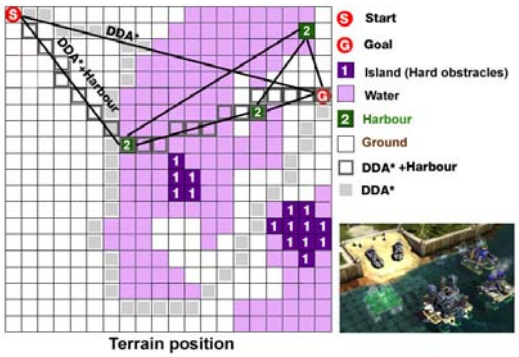


Fig.23. This is comparison between uses only DDA* which pathway more long than DDA* join in harbour station. http://files.playstatic.com/xbox-360/command-and-conquer-red-alert-3/thumbs/red-alert-3-screenshot-1.jpg

Fig.24. The state transition of pathway which has multiple transition layers. http://theirishgamers.files.wordpress.com/2008/08/ra3game-2008-08-04-16-48-20-59.jpg

The load game or reload file saved of games are finding whole pathways between every hurbours position which are posible. The whole connection generate abstract graph of pathways in every sea which use DDA* algorithm. It keeps in pathway database.

When player define begin position and destination position.

The first of step generate straight line between begin position and destination position.Next, It check straight line intersect with any water area and check set of that harbour position. The second of step compare distance between begin position to three harbours and choose shortest pathway for fixed harbour-in. The third of step compare distance between destination position to three harbours and choose shortest pathway for fixed harbour-out.The forth of step uses DDA* path finding to first harbour-in is nearest and retrieve information from pathway database for traverse to harbour-out. Next, it uses DDA* path finding to destination position or next harbour-in of next sea. Fifth, there are compare distance between abstarct graph in case use harbours and does not use harber and choose shortest path of pathway. We will get optimum pathway.[2] We may neglect this step cause decrease of speed. Finally, it operates follow pathway choosed.

G. The combination multilayer operation and auto transition terrain operation in strategy game, apply to unit.

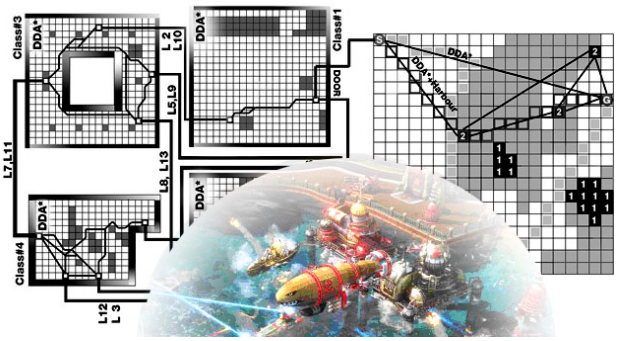


Fig.25. The combination of multiple transition layers and auto transition terrain. : http://pcmedia.ign.com/pc/image/article/923/923574/command-conquer-red-alert-3-20081024025255657_640w.jpg

The object of sky operate base on multilayer by use CLYi position together with database such as airport, flight station.

The object of sea operate base on auto-transition terrain find pathway between every harbours together with database.

The object of living operate base on both auto-transition terrain and multilayer together with pathway database such as man,animal and liveing .

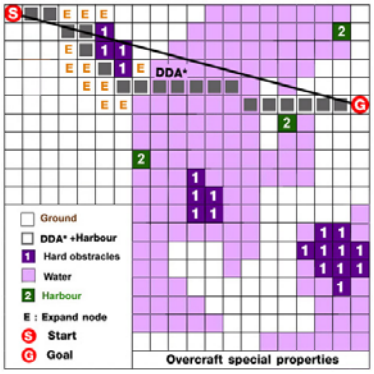


Fig.26.The properties of overcraft operate base on only DDA* algorithm.

The object of overcraft operate base only DDA* algorithm which movement capability both ground and sea. It does not change terrain type. It does not difference ground and sea.

The rest pathway use DDA* algorithm in realtime. The general pathway calculate shotest path DDA* in time together with abstract graph in path database which loading in first which update information after change in each part every time.It less update than stategy game kind god games.

Fig.27. The strategy game kind adventure style which scene less update than stategy game kind god games: http://z.about.com/d/compactiongames/1/0/k/F/commandos3k.jpg, http://www.deafgamers.com/commandos2_011.jpg

H. Proof of hypothesis.

Proof by principle hypothetical syllogism as follow:

$$P \to Q \text{ and } Q \to R \text{ so, } P \to R$$

The process of A* classic algotithm cause to any method which improve base on A* classic has efficiently. The DDA* has more efficient than A* classic. So, any method which improve base on DDA* algorithm deduce has more efficient than A* classic algorithm, too.

Proof by programming as follow:

This research tested with two programming which developed. First, the weak perspective projection model implement by Matlab. Second, AI pathfinding algorithms implement by VB 6.0 base on DirectX technology, Hardware CPU 2.8 Gb., memory 1 Gb. and compare speed with DirectX.TickCount function.

IV. Result of Experiment

From result of experiment, there are compare between various algorithms about take time and expand node.

TABLE I
ONE LAYER OF GROUND :OUTDOOR

Algorithm	Map 1# have not obstacles.		Map 2#	
	Take time(ms.)	Expand Node	Take time(ms.)	Expand Node
Dijkstra	109	783	109	643
BFS	46	133	31	123
DFS	63	397	47	217
IDA	15	133	16	113
A*	32	133	32	126
IDA*	31	133	31	126
Breadth FS	125	783	94	643
DepthD A*	2	27	16	90
Algorithm	**Map 3#**		**Map 4#**	
	Take time(ms.)	Expand Node	Take time(ms.)	Expand Node
Dijkstra	94	557	109	582
BFS	31	115	31	328
DFS	47	220	63	271
IDA	32	282	16	128
A*	32	108	31	129
IDA*	16	113	31	127
Breadth FS	94	555	94	582
DepthD A*	15	88	15	91

Fig.28. [Dijkstra] Dijkstra's algorithm, [BFS] Best First Search, [DFS] Depth First Search,[IDA] Iterative Deepening algorithm, [A*] A-Star Algorithm (A*),[IDA*] Iterative Deepening A* algorithm, [Breadth FS] Breadth First Search [***DepthD A****] Depth Direction A*

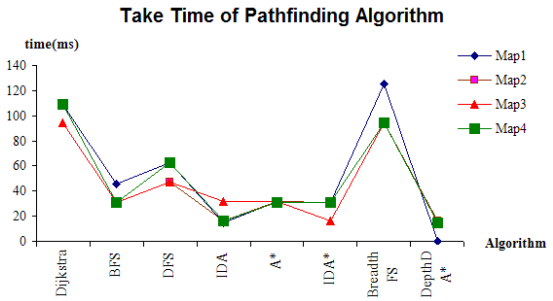


Fig.29.Take Time of Pathfinding Algorithm

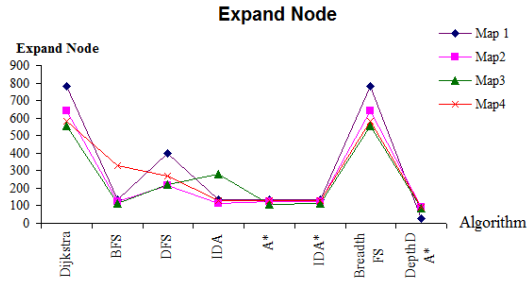


Fig.30.Expand Node of Pathfinding Algorithm

Experiment by Map in building: Class#1=10,000 tiles (block) Class#2=4820 tiles, Class#3=6700 tiles,Class#4=7400 tiles, Class#5=3750 tiles, Class#6=2400 tiles. Each class(layer) has 3 CLY positions.It connects between each class scatteredly.

In the building: Random hard obstracles 50% of area and CLY to 100 maps and on terrain cross sea 100 maps by start and goal tiles and does not pathway database.

Algorithm	In the building		On terrain encircle sea	
	Take time (ms.) (S)to(G) average	Expand Node average	Take time(ms.) average	Expand Node average
A*	(S1 to G6) 1286	4423	(T10000) 156	605
DepthD A*	(S1 to G6) 568	3151	(T10000) 79	433
A*	(S2 to G6) 861	3158	(T6400) 102	422
DepthD A*	(S2 to G6) 487	2253	(T6400) 51	280
A*	(S3 to G6) 647	2526	(T3600) 71	304
DepthD A*	(S3 to G6) 320	1869	(T3600) 42	224
A*	(S4 to G6) 428	2016	(T1600) 53	269
DepthD A*	(S4 to G6) 258	1447	(T1600) 27	137
Algorithm	**From inside building to terrain cross sea with harbour Combination multilayer & auto TST & Database**			
	(Start)to(Goal) average	Expand Node average	Take time(ms.) Expand Node	
A*	(S6 to T10000)	1469	5267	
DepthD A*	(S6 to T10000)	679	3616	
	(S5 to T6400)	994	3611	
DepthD A*	(S5 to T6400)	741	2566	
	(S4 to T3600)	762	2855	
DepthD A*	(S4 to T3600)	386	2120	
	(S3 to T1600)	527	2151	
DepthD A*	(S3 to T1600)	318	1619	

Remark :S[Class] to G[class]and T [tiles] .In experiment by CLY=1 per class,TST as auto-transition terrain, The experiment using pathway database.

V. Conclusion

The weak perspective projection model can represent to start position and destination position in game online of each player and able to scale position for real scene. It sent position of a player with two points, beside it can set start position and destination position from server to client although different layer. And in client IA will working every time. *The pathway database keeps abstract graph of each building and abstract graph of each harbour.* The DDA* algorithm decrease expand child node 50.12-58.67% depend on property of terrain and object propeterty. The DDA* together with pathway database decrease expand child node 69.12-70.67% and increase speed to 36-46% of A*. It follows as hypothesis of this research.The DDA* increase speed and smooth of pathway and use pathway database up speed extremely.

I hope DDA* effect to any method improve base on A* classic algorithm which DDA* are better.

References

[1] Daniel Harabor, Adi Botea."Hierachical Path Planning for Multi-Size Agents in Herterogeneous Environment". Perth, Australia. IEEE Symposium on Computational Intelligence and Games (CIG'08), pp.258-264, Dec 15-18, 2008

[2] A. Botea, M. Müller, J. Schaeffer. "Near Optimal Hierarchical Path-Finding" Journal of Game Development (2004) Volume 1, Issue1 7-28.

[3] T. C. H. John, E. C. Prakash, and N. S. Chaudhari, "Team AI: probabilistic pathfinding," in Proceedings of the International Conference on Game Research and Development, vol. 223 of ACM International Conference Proceeding, pp. 191–198, Perth, Australia, December 2006.

[4] M. Buckland, Programming Game AI by Example, Wordware, Plano, Tex, USA, 2005